LexEVS 6.0 CTS2 Authoring 4 - Usage Context Authoring Operation API

Contents of this Page

CTS2 Links for LexEVS 6.0

- CTS2 API Main Page
- Programmer's Guide Main Page
- LexEVS 6.0 Main Page
- LexEVS Current Release

Introduction

LexEVS CTS2 Usage Context Authoring API provides capability to author Usage Context and also the ability to create a Code System that can hold the Usage Contexts.

Here are the authoring functions that can be performed on Usage Context:

- **Create** - This function provides capability to create:
  - New Code System to hold cUsage Contexts
  - New Usage Context
  - Add new property to Usage Context
- **Edit** - This function provides capability to update:
  - Property of a Usage Context
- **Remove** - This function provides capability to remove:
  - Usage Context
  - Property of a Usage Context
- **Versionable Change** - This function provides capability to modify versionable attributes of Usage Context like Status, Effective Date, Expiration Date, isActive and Owner:
  - Usage Context Status
  - Activate Usage Context
  - De-Activate Usage Context

Interface

org.lexevs.cts2.author.UsageContextAuthoringOperation is the main interface for all the authoring operations against Usage Context. This interface can be accessed using main LexEVSCTS2 interface, like:

```
org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthOp = new org.lexevs.cts2.LexEvsCTS2Impl().
getAuthoringOperation().getUsageContextAuthoringOperation();
```

Revision Information

All the authoring functions described here requires information about the author and revision/version id to be assigned to entities for each of these operations. These is done passing object **org.lexevs.cts2.core.update.RevisionInfo**. RevisionInfo object has following attributes:

- **java.lang.String changeAgent** - (Optional) The source that participated in this particular change.
- **java.lang.String changeInstruction** - (Optional) A human or machine readable set of instructions on how to apply this change.
- **java.lang.String revisionId** - (**Mandatory**) The unique identifier of this revision.
- **java.lang.Long editOrder** - (Optional) The relative order that this revision is to be applied if in a systemRelease.
- **java.util.Date revisionDate** - (Optional) The end date for which this version is operative (considered commited).
- **java.lang.String description** - (Optional) The description of the resource/change.
- **java.lang.String systemReleaseURI** - (Optional) The official URI of this release

Authoring Functions

Following sections contains detailed functions provided by UsageContextAuthoringOperation interface.

Create Operations

Create Operation provides capability to create a Code System to hold Usage Contexts, Usage Context and Properties. For every entry that gets created, a unique revision(version) identifier will be assigned to that entry. This helps in retrieving snapshots of an entry based on its revision(version) identifier. This unique id can be passed in using the RevisionInfo object described above.

createUsageContextCodeSystem

This function provides capability to create a new Code System to contain a set of Usage Contexts. The Code System is created by defining the set of meta-data properties that describe it. At this point there is no Usage Context added.

```
createUsageContextCodeSystem(RevisionInfo revision, String codeSystemName, String codeSystemURI, String formalName, String defaultLanguage, long approxNumConcepts, String representsVersion, List<String> localNameList, List<org.LexGrid.commonTypes.Source> sourceList, Text copyright, Mappings mappings)
```

Description:	Creates new code system to hold Usage Contexts.
Input:	• ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc. • ***java.lang.String codingSchemeName*** - (**Mandatory**) Usage Context Code System Name. • ***java.lang.String codingSchemeURI*** - (**Mandatory**) Usage Context Code System URI. • ***java.lang.String formalName*** - (Optional) Formal name of a Usage Context Code System. • ***java.lang.String defaultLanguage*** - (Optional) Default language of Usage Context Code System. • ***long approxNumConcepts*** - (Optional) Approximate number of Usage Contexts this Usage Context Code System may contain. • ***java.lang.String representsVersion*** - (**Mandatory**) Initial version of the Usage Context Code System. • ***java.util.List<java.lang.String> localNameList*** - (Optional) Any local name(s)/reference(s) for this Usage Context Code System used within the Code System. • ***java.util.List<org.LexGrid.commonTypes.Source> sourceList*** - (Optional) Source(s) of this Usage Context Code System. • ***org.LexGrid.commonTypes.Text copyright*** - (Optional) Information about rights held in and over the Usage Context Code System. Typically, copyright information includes a statement about various property rights associated with the Usage Context Code System, including intellectual property rights. • ***org.LexGrid.naming.Mappings mappings*** - (**Mandatory**) A list of all of the local identifiers and defining URI's that are used in the Usage Context Code System.
Output:	***org.LexGrid.codingSchemes.CodingScheme*** - Created Usage Context Code System
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* Populate new Usage Context code system meta data:
	`String codingSchemeURI = "urn:oid:11.11.22.999";`
	`String representsVersion = "1.0";`
	`String codingSchemeName = "Usage Context Coding Scheme";`
	`String formalName = "CTS 2 API Created Usage Context Code System";`
	`String defaultLanguage = "en";`
	`Long approxNumConcepts = new Long(1);`
	`List<String> localNameList = Arrays.asList();`
	`org.LexGrid.commonTypes.Source source = new org.LexGrid.commonTypes.Source();`
	`source.setContent("source");`
	`List<org.LexGrid.commonTypes.Source> sourceList = Arrays.asList(source);`
	`Text copyright = new Text();`
	`org.LexGrid.naming.Mappings mappings = new org.LexGrid.naming.Mappings();`
	`org.LexGrid.naming.SupportedLanguage supportedLang = new org.LexGrid.naming.SupportedLanguage();`
	`supportedLang.setLocalId("en");`
	`supportedLang.setUri("URI_for_lang_en");`
	`mappings.addSupportedLanguage(supportedLang);`
	• *Step 4:* call create method to create the Usage Context code system:
	`CodingScheme codeScheme = ucAuthorOp.createUsageContextCodeSystem(revInfo, codingSchemeName, codingSchemeURI, formalName, defaultLanguage, approxNumConcepts, representsVersion, localNameList, sourceList, copyright, mappings);`

createUsageContext

This function creates a Usage Context to be included in a Code System. The new Usage Context is defined by the set of meta-data properties that describe it.

`createUsageContext(String usageContextId, String usageContextName, String namespace, RevisionInfo revisionInfo, String description, String status, boolean isActive, Properties properties, String codeSystemNameOrURI, String codeSystemVersion)`

Description:	Creates new Usage Context in a code system.

Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) ID of a new Usage Context. • ***java.lang.String usageContextName*** - (**Mandatory**) Name of a new Usage Context. • ***java.lang.String namespace*** - (**Mandatory**) Namespace of a new Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc. • ***java.lang.String description*** - (**Mandatory**) Description of a new Usage Context. • ***java.lang.String status*** - (Optional) Status of new Usage Context. • ***boolean isActive*** - (Optional) Status of new Usage Context. • ***org.LexGrid.commonTypes.Properties*** - (Optional) List of properties for the new Usage Context. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of a Code System that will hold this new Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of a Code System that will hold this new Usage Context.
Output:	***java.lang.String*** - Usage Context id if created successfully
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* Populate a properties to be added:
	`Property prop = new Property();`
	`prop.setPropertyId("propertyId1");`
	`prop.setPropertyName("propertyName");`
	`prop.setIsActive(false);`
	`prop.setLanguage("en");`
	`prop.setOwner("owner");`
	`prop.setPropertyType(PropertyTypes.PROPERTY.name());`
	`Text text = new Text();`
	`text.setContent("content");`
	`text.setDataType("Text datatype");`
	`prop.setValue(text);`
	`Properties props = new Properties();`
	`props.addProperty(prop);`
	• *Step 3:* call create Usage Context method by passing code system version and Usage Context information:
	`String usageContextId = ucAuthOp.createUsageContext("UC00A", "Automobiles-CD", "Automobiles", revInfo, "Usage Context for automobiles", "pending", false, props, "Automobile", "1.0",);`

addUsageContextProperty

This function provides capability to add a new property to a Usage Context.

`addUsageContextProperty(String usageContextId, String namespace, Property newProperty, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Add new property for a Usage Context.

Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) Identifier of a Usage Context to which a new property will be added. • ***java.lang.String namespace*** - (**Mandatory**) Namespace of a Usage Context to which a new property will be added. • ***org.LexGrid.commonTypes.Property newProperty*** - (**Mandatory**) New property that will be added to the Usage Context. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of a Code System that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of a Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if addition of new property was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* Populate a property to be added:
	`Property prop = new Property();`
	`prop.setPropertyId("propertyId1");`
	`prop.setPropertyName("propertyName");`
	`prop.setIsActive(false);`
	`prop.setLanguage("en");`
	`prop.setOwner("owner");`
	`prop.setPropertyType(PropertyTypes.PROPERTY.name());`
	`Text text = new Text();`
	`text.setContent("content");`
	`text.setDataType("Text datatype");`
	`prop.setValue(text);`
	• *Step 4:* call add property method by passing the code system information, Usage Context information and a new property:
	`boolean added = ucAuthOp.addNewUsageContextProperty("UC00A", "Automobiles-UC", prop, "Automobiles", "1.0", revInfo);`

Edit Operations

The edit operation provides the capability to modify properties of a Usage Context. For every entry that gets modified, a unique revision(version) identifier will be assigned to that entry. This helps in retrieving snapshots of an entry based on its revision(version) identifier. This unique id can be passed in using the RevisionInfo object described above.

updateUsageContextProperty

This function provides capability to modify existing property of a Usage Context.

`updateUsageContextProperty(String UsageContextId, String namespace, Property changedProperty, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Modifies existing property of a Usage Context.

Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) Identifier of a Usage Context that contains the property. • ***java.lang.String namespace*** - (**Mandatory**) Namespace of a Usage Context that contains the property. • ***org.LexGrid.commonTypes.Property property*** - (**Mandatory**) Modified property. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of a Code System that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of a Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if update was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* Populate a modified property. Just modified isActive to 'true' and language to 'eng' in property 'propertyId1':
	`Property prop = new Property();`
	`prop.setPropertyId("propertyId1");`
	`prop.setPropertyName("propertyName");`
	`prop.setIsActive(true);`
	`prop.setLanguage("eng");`
	• *Step 4:* call update property method by passing the code system , Usage Context information and the modified property:
	`boolean updated = ucAuthOp.updateUsageContextProperty("UC0001", "Automobiles-UC", prop, "Automobiles", "1.0", revInfo); |`

Remove Operations

Remove operation provides capability to remove Usage Context and its property.

removeUsageContext

This function provides capability to remove a Usage Context.

`removeUsageContext(String usageContextId, String namespace, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Removes a Usage Context from code system.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) Identifier of a Usage Context to be removed. • ***java.lang.String namespace*** - (**Mandatory**) Namespace of a Usage Context to be removed. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of a Code System that contains the Usage Context to be removed. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of a Code System that contains the Usage Context to be removed. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if remove was success

Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation cdAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* call remove Usage Context method by passing the code system and Usage Context information:
	`boolean removed = ucAuthOp.removeUsageContext("UC0001", "Automobiles-UC", "urn:oid:11.11.0.99", "1.0", revInfo);`

removeUsageContextProperty

This function provides capability to remove property of a Usage Context.

`removeUsageContextProperty(String usageContextId, String namespace, Property property, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Removes a property of a Usage Context.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) Usage Context identifier that contains the property. • ***java.lang.String namespace*** - (**Mandatory**) Namespace of a Usage Context that contains the property. • ***org.LexGrid.commonType.Property property*** - (**Mandatory**) Property that needs to be removed. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Code System URI/name that contains the Usage Context. • ***java.lang.String representsVersion*** - (**Mandatory**) Version of the Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if remove was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 3:* Populate property that needs to be removed:
	`Property propertyToRemove = new Property();`
	`propertyToRemove.setPropertyId("p1");`
	• *Step 4:* Call remove property method by passing the code system, Usage Context and property information:
	`boolean removed = ucAuthOp.deleteUsageContextProperty("UC0001", "Automobiles-UC", propertyToRemove, "urn:oid:11.11.0.99", "1.0", revInfo);`

Versionable Change Operations

Versionable Change operation provides capability to modify versionable attributes of Usage Context like Status, Effective Date, Expiration Date, isActive and Owner.

updateUsageContextStatus

This function modifies the status of a Usage Context.

`updateUsageContextStatus(String usageContextId, String namespace, String newStatus, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Modifies the status of a Usage Context.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) identifier of a Usage Context. • ***java.lang.String namespace*** - (**Mandatory**) Mamespace of a Usage Context. • ***java.lang.String status*** - (**Mandatory**) Modified status value. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Code System URI that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of the Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if update was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 2:* Call status change method by passing the code system, Usage Context and status information:
	`boolean changed = ucAuthOp.updateUsageContextStatus("UC00001", "Automobiles-UC", "Active", "urn:oid:11.11.0.99", "1.0", revInfo);`

activateUsageContext

This function activates Usage Context so that it can be accessed in the terminology service.

`activateUsageContext(String usageContextId, String namespace, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Activates Usage Context.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) identifier of a Usage Context. • ***java.lang.String namespace*** - (**Mandatory**) Mamespace of a Usage Context. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Code System URI that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of the Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if activation was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 2:* Call activate method by passing the code system and Usage Context information:
	`boolean activated = ucAuthOp.activateUsageContext("UC00001", "Automobiles-UC", "urn:oid:11.11.0.99", "1.0", revInfo);`

deactivateUsageContext

This function deactivates Usage Context so that it can no longer be accessed in the terminology service.

`deactivateUsageContext(String usageContextId, String namespace, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Deactivates UsageContext.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) identifier of a Usage Context. • ***java.lang.String namespace*** - (**Mandatory**) Mamespace of a Usage Context. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Code System URI that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of the Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if deactivation was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:
	`org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();`
	• *Step 2:* Populate RevisionInfo object:
	`RevisionInfo revInfo = new RevisionInfo();`
	`revInfo.setChangeAgent("change Agent Name");`
	`revInfo.setChangeInstruction("here goes the change Instructions");`
	`revInfo.setDescription("description of the resource");`
	`revInfo.setEditOrder(1L);`
	`revInfo.setRevisionDate(new Date());`
	`revInfo.setRevisionId(UUID.randomUUID().toString());`
	• *Step 2:* Call deactivate method by passing the code system and Usage Context information:
	`boolean deactivated = ucAuthOp.deactivateUsageContext("UC00001", "Automobiles-UC", "urn:oid:11.11.0.99", "1.0", revInfo);`

updateUsageContextVersionable

This function provides capability to modify Usage Context versionable attributes like effective date, expiration date, owner, status etc.

`updateUsageContextVersionable(String usageContextId, String namespace, Versionable changedVersionable, String codeSystemNameOrURI, String codeSystemVersion, RevisionInfo revisionInfo)`

Description:	Update Usage Context versionable attributes like effective date, expiration date, owner, status etc.
Input:	• ***java.lang.String usageContextId*** - (**Mandatory**) identifier of a Usage Context. • ***java.lang.String namespace*** - (**Mandatory**) Mamespace of a Usage Context. • ***org.LexGrid.commonTypes.Versionable changedVersionable*** - (**Mandatory**) versionable (like:owner, effectiveDate, expirationDate, status etc) changes. • ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Code System URI that contains the Usage Context. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of the Code System that contains the Usage Context. • ***org.lexevs.cts2.core.update.RevisionInfo revision*** - (**Mandatory**) Contains revision information like unique RevisionId, change description, author information etc.
Output:	***boolean*** - True; if update was success
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:

- *Step 1:* Instantiate UsageContextAuthoringOperation if it is not done yet:

```
org.lexevs.cts2.author.UsageContextAuthoringOperation ucAuthorOp = LexEvsCTS2Impl.
defaultInstance().getAuthoringOperation().getUsageContextAuthoringOperation();
```

- *Step 2:* Populate RevisionInfo object:

```
RevisionInfo revInfo = new RevisionInfo();
revInfo.setChangeAgent("change Agent Name");
revInfo.setChangeInstruction("here goes the change Instructions");
revInfo.setDescription("description of the resource");
revInfo.setEditOrder(1L);
revInfo.setRevisionDate(new Date());
revInfo.setRevisionId(UUID.randomUUID().toString());
```

- *Step 2:* Populate versionable changes:

```
Versionable changedVersionable = new Versionable();
changedVersionable.setEffectiveDate(new Date());
changedVersionable.setIsActive(true);
changedVersionable.setOwner("new Owner");
changedVersionable.setStatus("new status");
```

- *Step 3:* Call change versionable method by passing the code system, Usage Context and changed versionable information:

```
boolean changed = ucAuthOp.updateUsageContextVersionable("UC00001", "Automobiles-UC",
changedVersionable, "urn:oid:11.11.0.99", "1.0", revInfo);
```